ABOVE THE GROUND PLANE

Reception vs. SDR Hardware

In his last article, Ed presented the analog parts of a WWVB receiver that he was working on: the antenna and preamplifier that extract a useful signal level from the air. This month, starting with a resonant loop antenna and a tuned preamplifier, Ed combines SDR hardware with Raspberry Pi 3 software to explore short-term variations in groundwave propagation.

By Ed Nisley

COLUMNS

The ideal software-defined radio (DSR) would digitize the RF voltage directly at the antenna terminals and pass those numbers to algorithms that amplify, filter, and demodulate them into a useful signal. Even an ideal SDR requires enough RF energy to extract the desired signal from noise, which depends on the attenuation between the transmitter and receiver: the most advanced algorithms can't make up for a nonexistent signal.

PHOTO 1
The SMA adapter cable lashed to the NooElec NESDR Mini 2+ (upper right) protects its delicate MCX connector. The software-compatible ThumbNet N3 SDR (lower right) has an improved PCB layout for lower RF noise, better thermal control, and a rugged F input connector. The black-box SDRplay RSP1 (left) offers better overall performance, but requires a closed-source driver.

The resonant loop antenna I described in my March column converts the magnetic component of the WWVB signal into a differential voltage much too small for direct conversion by any practical ADC. An instrumentation amplifier and bandpass filter boost and isolate the signal, producing enough voltage to be useful, even with the non-ideal SDRs on my bench.

In this column, I'll describe some of the SDR hardware I've tried, show some RF signal (and noise!) spectra, and explore short-term variations affecting the RF arriving at the receivers.

RADIO HARDWARE

The hardware in **Photo 1** shows what happens as software replaces hardware: radios become small, plain boxes with only RF inputs, data outputs, and perhaps a power connector. In fact, a white LED in the NooElec radio provides the only hardware "user interface" in that collection.

All of the radios pass data and control through their USB 2.0 ports, each with a different connector. The NooElec USB Type-A plug fits directly into a downstream receptacle on a host system, just like a USB memory device, but its large case may interfere with adjacent USB devices. In contrast, the SDRplay radio has a standard Type-B receptacle and the ThumbNet radio has a mini Type-B receptacle. While I was testing these radios, my desktop PC sprouted three different USB cables.

Diminutive radios call for correspondingly small RF input connectors, again with little standardization, although the SMA connector on the SDRplay seems to be appearing on everything other than the least-expensive radios. The snap-fit MCX connector on the NooElec NESDR 2+ mates with the cable from its accompanying antenna and requires an adapter for any other RF hardware. The ThumbNet N3's television-style F connector provides compatibility with their existing satellite ground station hardware; the input's 75 Ω impedance introduces a 1.5:1 SWR mismatch with 50 Ω hardware that probably won't matter with most antennas and preamplifiers.

All three radios are receivers, not transceivers, and can draw power directly from the USB port's +5 VDC supply, even though USB power is notorious for poor stability and terrible EMI. The radios have maximum current requirement approaching the port's 500 mA upper limit and require a high-quality USB cable to reduce the voltage drop. The ThumbNet N3 features a separate power connector for an external +5 V DC supply, which should have a linear regulator for low noise.

Both the NooElec and ThumbNet radios have RT820 tuner and RTL2832 receiver/controller chipsets, a temperature-compensated crystal oscillator, and an 8-bit ADC for a 48 dB dynamic range. The SDRplay radio has a 12-bit ADC with 10 useful bits and 60 dB of effective dynamic range. All three have RF preamplifiers that boost both signal and noise levels. They also offer hardware AGC (automatic gain control), which, as you'll see, isn't particularly helpful with very narrowband signals.

RT820 tuners operate from 25 to 1750 MHz, with slight variations on each end due to voltage and thermal issues. The SDRplay has a lower limit of 100 kHz (some documents say 10 kHz) and an upper limit of 2 GHz, with automatic preselection filters to reduce RF energy far from the tuned frequency. None of the radios can directly tune the 60 kHz WWVB signal.

As far as costs go, the NooElec radio is about $25, units without the stabilized oscillator go for $17, and other vendors offer no-name units with allegedly similar circuitry for less than $10. Proceeds from the $35 ThumbNet radio support their STEM satellite project; you can get the radio board alone for $25, plus $10 for an internal EMI shield and aluminum case. The SDRplay RSP1 runs about $150, although I bought mine on sale when they introduced the RSP2 models.

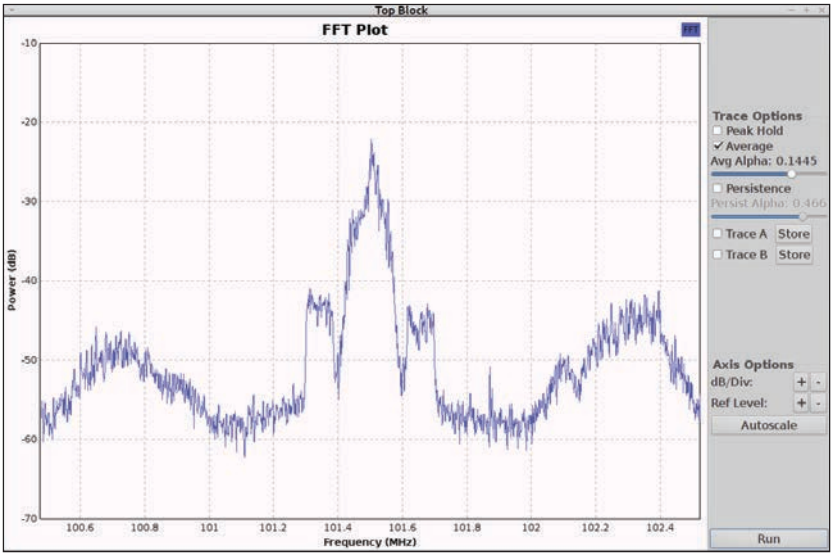

PHOTO 2
The RF spectrum of an FM broadcast station shows the analog carrier rising above the background noise, with digital data sidebands extending on either side. Distant FM stations produce the humps ±800 kHz from the carrier.

Photo 2 shows a simple way to verify a new radio with a local FM broadcast station providing a test signal. I connected an

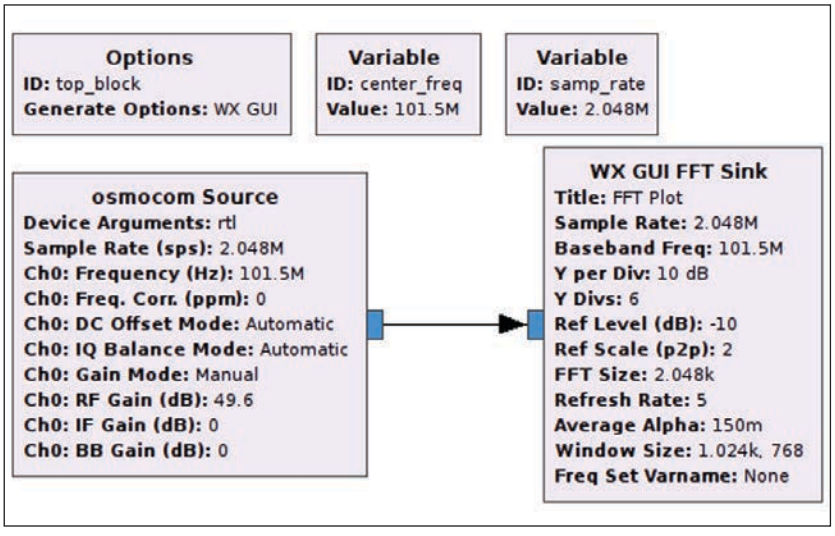

FIGURE 1
This GNU Radio flow diagram sets the SDR tuner to 101.5 MHz, then converts the sampled data into the FFT spectrum display appearing in Photo 2.

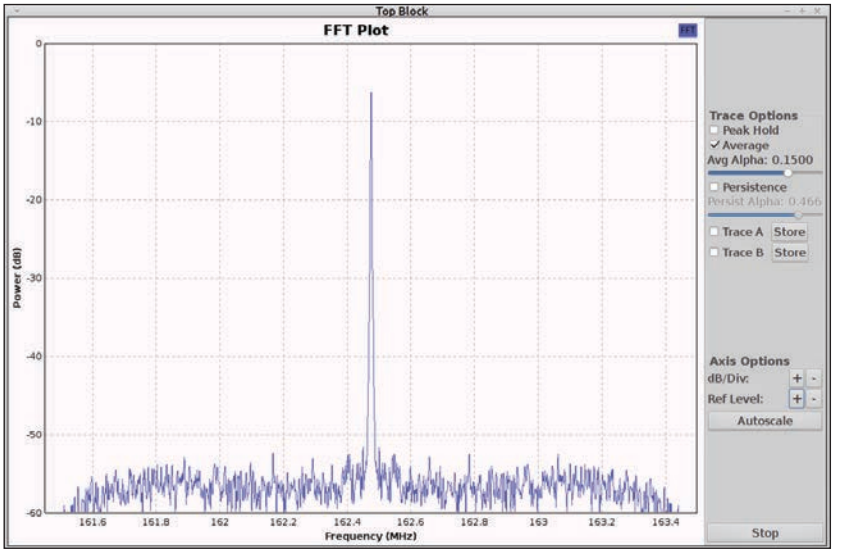

PHOTO 3

The narrow spectrum of a National Weather Service station corresponds to the low bandwidth required for a monaural voice without any accompanying in-band signals or data.

amateur radio VHF-UHF dual-band antenna to the NooElec radio through a series of adapters reducing the antenna's UHF connector to BNC to SMA to the SMA-to-MCX cable in Photo 1. Although WPDH's 101.5 MHz carrier frequency lies below the amateur radio VHF band (144 to 148 MHz), their 4.4 kW ERP more than compensates for any losses.

The two-block GNU Radio flow graph in **Figure 1** produced Photo 2. The `osmocom Source` block tunes the radio to 101.5 MHz and sets the sampling rate to 2.048 MHz, while the `FFT Sink` block displays the spectrum resulting from Fast Fourier Transforms applied to successive blocks, each containing 2048 samples, arriving every millisecond.

The central peak in Photo 2 corresponds to the station's analog stereo FM signal, with a 200 kHz total bandwidth; at this resolution, you cannot see the analog subcarriers and pilot tones. The relatively flat shoulders on either side represent the digital data carrying the station's HD Radio audio channels, with the total bandwidth sharply limited to ±200 kHz around the analog carrier. The gentle humps centered on 100.7 and 102.3 MHz come from "over the horizon" FM stations in adjacent states.

If you're in the continental US, you can tune your SDR to a nearby National Weather Service station for a narrowband FM test signal: the spike in **Photo 3** shows the spectrum of WXL-37 at 162.475 MHz. That frequency lies above the antenna's VHF band, but a 1 kW transmitter gets the signal through. Obviously, a synthesized monaural voice occupies far less RF bandwidth than the multiple audio and data streams of an "album oriented" rock music station. Both of those FM stations broadcast from antennas bolted to the same tower on Illinois Mountain, directly across the Hudson River from Poughkeepsie, about 7.6 line-of-sight miles = 12.2 km from our house.

WWVB VS. LOCAL OSCILLATORS

Although receiving a strong VHF FM signal from a nearby transmitter suffices to demonstrate that an SDR works, the LF signal from WWVB's transmitter halfway across the continent reveals a different set of problems. **Photo 4** shows the hardware I used for the rest of the column, lined up on the plywood floor of our attic, as far from household EMI sources as I could manage.

In my March 2017 column, I described how to convert a ribbon cable into a 40 turn magnetic loop antenna; the white 3D printed block reinforces the solder joints and stabilizes the three-wire preamplifier cable. The ribbon hangs in a lumpy circle from screws driven into the rafters, taking advantage of our house's alignment to aim the loop plane westward toward the WWVB transmitter in Colorado.

The hand-wired preamplifier circuit board on the left of Photo 4, also described in March, holds the mica compression capacitors forming a tank circuit with the loop inductance. Adjusting those capacitors tunes the tank resonance to 60 kHz. An instrumentation amplifier converts the antenna's differential RF signal to a ground-referenced voltage for

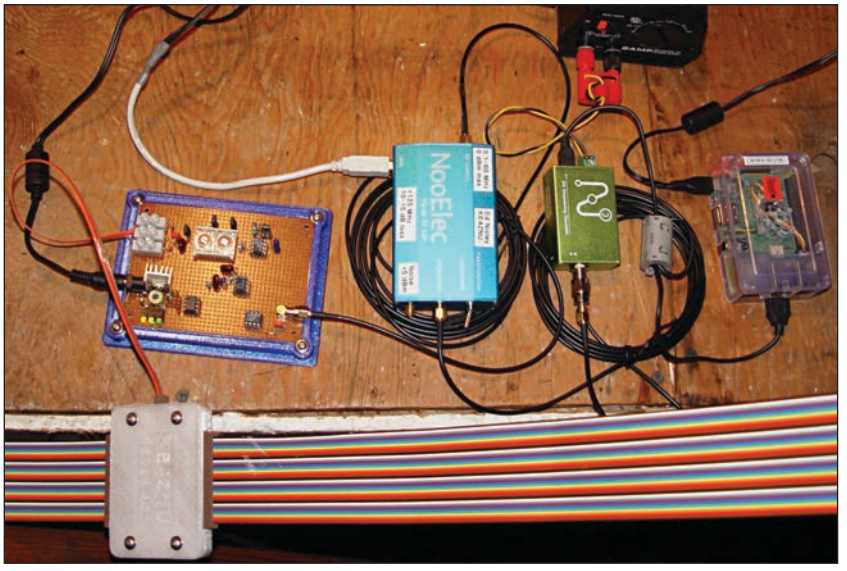

PHOTO 4

The WWVB receiver components on my attic floor resemble a block diagram, with the loop antenna along the bottom feeding the 60 kHz tuned preamplifier on the left, a NooElec 125 MHz upconverter, the N3 SDR radio, and a Raspberry Pi 3. The loop antenna plane points west-southwest, toward the NIST WWVB transmitter in Colorado.

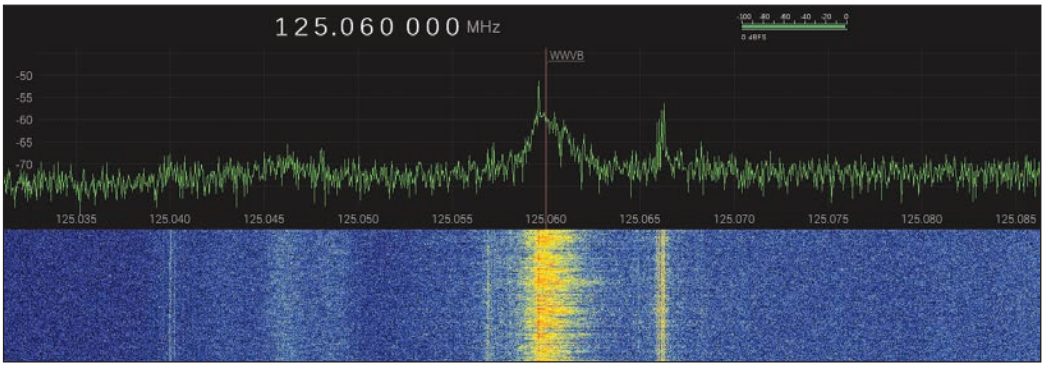

PHOTO 5
The WWVB signal rises barely 10 dB above the noise level within the preamplifier's passband around 60 kHz, translated to 125.060 MHz; both the displayed frequency and the loop antenna resonance show the need for slight tweaking. Note the horizontal scale change compared with the FM broadcast signals in Photos 2 and 3.

a bandpass filter. Overall, the antenna and preamplifier have a 2 kHz bandwidth.

The linear power supply just above the ThumbNet N3 provides clean +4.5 VDC at 420 mA, which the N3 uses in preference to USB power. Its case runs 17°C warmer than the attic's 7°C while dissipating just under 2 W: I should relocate the electronics to the bottom of the attic stairwell during the summer.

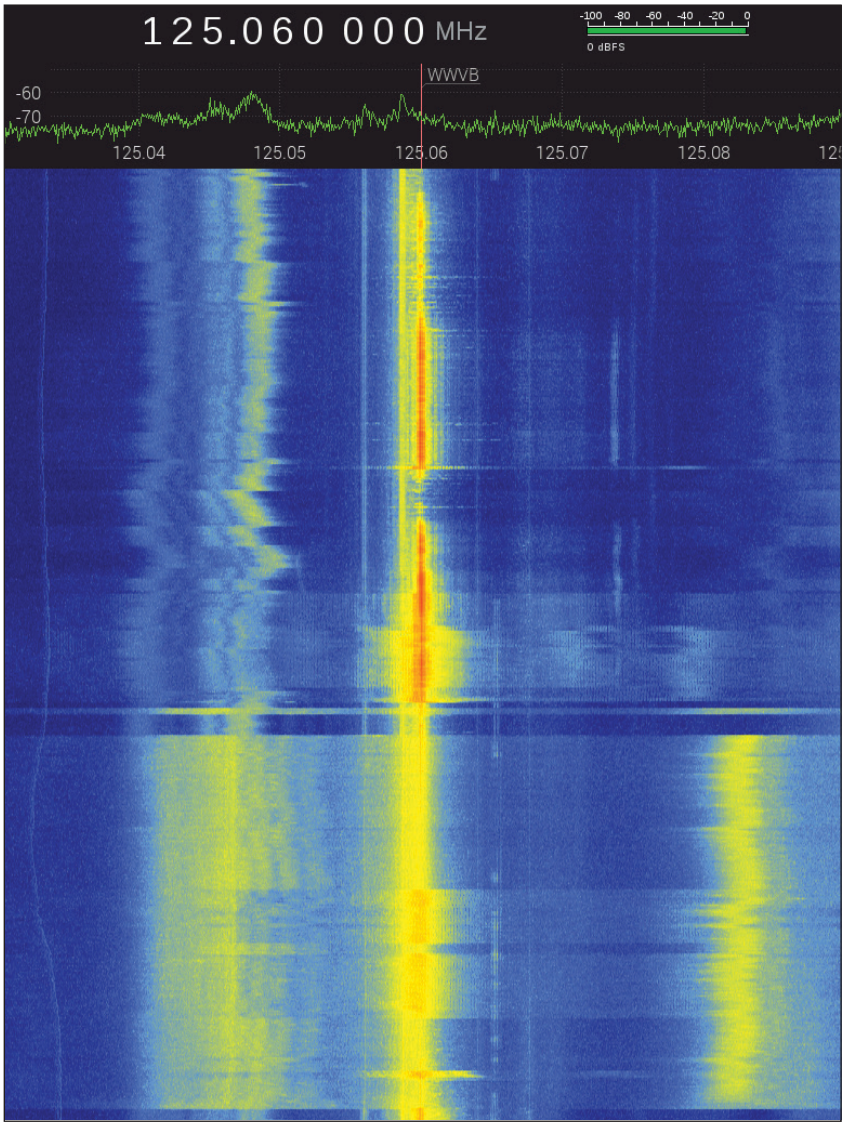

PHOTO 6
The RF spectrum around 60 kHz contains a wide variety of spurious signals that change both abruptly and erratically over the course of 24 hours. The birdie along the left edge seems to vary inversely with ambient temperature.

On the right side of Photo 4, the N3 plugs into a headless Raspberry Pi 3 through a short USB cable sporting a ferrite core to reduce EMI. The Pi connects to our Wi-Fi network using its internal bar antenna. I access its Raspbian GUI through VNC and use a command-line SSH session for system maintenance, so there's no need for a display, keyboard, and mouse in the attic.

None of the SDRs in Photo 1 can tune to the 60 kHz WWVB carrier frequency, so the NooElec Ham-It-Up mixer at the center of Photo 4 translates the analog RF signal upward by 125 MHz, putting it well inside the SDR's normal tuning range. The Ham-It-Up has a lower frequency bound of 100 kHz, with a nominal 10 dB conversion loss across its 60 MHz passband. I measured 14 dB of loss at 60 kHz, so more more preamplifier gain may be in order.

Photo 5 shows the spectrum around the WWVB carrier, after the NooElec mixer translates it to 125.060 MHz. The screenshot comes from the GQRX receiver program running on the Raspberry Pi, relayed to my desktop monitor through VNC.

The tiny spike just to the left of the red cursor at the center represents the 60.000 kHz WWVB signal at maximum power output. With the transmitter oscillator locked to one of the most precisely known frequency standards in the world, slight inaccuracies in the local oscillators in my attic produce the small offset from 125.060 MHz: by definition, the transmitter end of the link has no frequency error!

Because the WWVB time code modulates the carrier amplitude and phase at about 1 Hz, the signal bandwidth is essentially 1 Hz. In comparison with those FM broadcast signals, WWVB will always look like a spike. In fact, the FFT frequency resolution will determine the width of the signal in SDR radio displays.

During my tests, various combinations of hardware produced frequency offsets ranging

from the –3 ppm shown in Photo 5 to about +13 ppm. The upconverter and SDR oscillators have 0.5 ppm temperature stability, but they're still in the early parts of their operational lives and may experience some frequency changes as they settle down. I did not correct the offsets in the measurements for this column, not least because it's easier to see the WWVB signal when it's not overlaid by a cursor.

The small hump around the WWVB carrier represents the loop antenna and preamplifier passband, with the –3 dB bandwidth of 2 kHz across the top. The height of the hump suggests an overall gain of 15 dB above the noise floor, half of the 30 dB I measured with a clean RF input from a signal generator.

The WWVB carrier rises from the left side of the passband hump, rather than its center, because the loop antenna resonance is slightly too high. As I showed in the March column, the loop's inductance varies with the shape of the loop and the surrounding materials, so each loop configuration requires fine-tuning the mica compression capacitors on the preamp circuit board to set the proper bandpass center frequency. I didn't adjust the loop resonance for the measurements you'll see in this column.

The waterfall display in **Photo 6** shows the RF environment around the WWVB signal over the course of 24 hours, starting and ending at 0800 local time. The colorful stripe down the middle shows the preamplifier passband, with orange and red marking the strongest signals, and a line from the WWVB carrier sometimes visible amid the clutter.

The upper half of the waterfall shows the relatively strong WWVB signal during overnight "dark path" conditions between Colorado and New York. The lower half shows how increasing attenuation reduces the signal during daylight hours: unlike those FM broadcast signals, WWVB fades out completely during much of the day and noise fills the preamp bandpass.

The thin line near the left edge of the waterfall is a birdie from an unknown local source. It seems to vary inversely with temperature and may be a mixing product created by hardware in the attic, rather than an actual oscillator. Even though the Raspberry Pi board complies with EMI suppression specifications, slight leakage from its myriad digital PCB traces can induce weak signals in nearby hardware.

The upconverter, SDR, and Pi oscillators contribute to the slight frequency offset mentioned earlier, but remain stable throughout the entire 24-hour waterfall. Any variation would affect the traces of the WWVB carrier, the filter bandwidth, and the

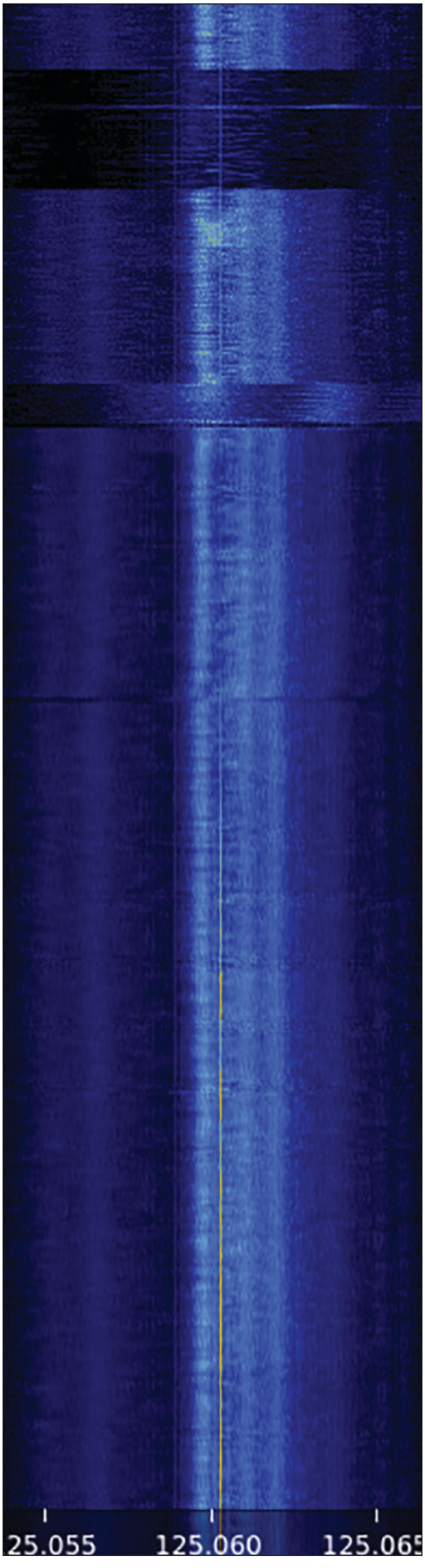

PHOTO 7
A high-resolution FFT shows the WWVB carrier fading away at dawn, near the middle of this 8 hour waterfall, then returning a few hours later. The bright blue band marking the preamplifier passband isn't quite centered at 60 kHz and the receiver's frequency requires a –2 ppm correction.

ABOUT THE AUTHOR

Ed Nisley is an EE and author in Poughkeepsie, NY. Contact him at ed.nisley@pobox.com with "Circuit Cellar" in the subject line to avoid spam filters.

many external sources, all of which appear as bar-straight lines showing the constant frequencies of those oscillators.

SIGNAL VARIATIONS

The horizontal stripes in Photo 6 represent the radio's response to external signals far outside the frequency range appearing in the waterfall. Other records show even more dramatic variations in the overall signal level, with instantaneous changes across the entire waterfall.

Activating the radio's internal AGC hardware increases that problem, because a strong carrier appearing or disappearing anywhere within the SDR's tuning range will affect the receiver gain. In fact, the AGC will produce the same result when it reacts to broadband noise by adjusting the receiver gain: the desired signal gets stronger or weaker as the noise level falls or rises.

With the AGC switched off, the waterfall in **Photo 7** shows much more stable conditions during seven hours of a January morning from 0350 to 1050. The WWVB carrier forms the single-pixel yellow line; I did not apply the –2 ppm correction required to align it with the 125.060 MHz tick mark. The blue band represents the preamplifier's passband, which, as before, isn't quite centered on 60 kHz.

Even at this tight frequency zoom, the WWVB signal's 1 Hz bandwidth produces a very narrow trace. In fact, the waterfall comes from a 524288 point FFT with 3.4 Hz resolution, so the WWVB signal can fit entirely within a single FFT frequency bin.

The horizontal stripes near the top of the waterfall demonstrate that, even with AGC turned off, the receiver's gain still varies slightly as strong signals appear and disappear elsewhere in the tuning range. A close look shows the WWVB carrier remains present, albeit with step-change amplitude variations within the stripes, so the effect originates in the receiver, not the WWVB signal.

A NOT-SO-SIMPLE MATTER OF SOFTWARE

Much of the software behind a "software defined radio" depends on the vectorized floating-point instructions built into contemporary CPUs and GPUs, which means that installing and using the GNU Radio SDR software on desktop PCs with x86_64 hardware poses little challenge. The same can't be said for the Raspberry Pi, because its ARM-based RISC CPU was optimized for low power rather than high performance.

The radios I'm using sample the RF signal at rates between about 1 MHz and 3 MHz, with each sample containing two numbers representing the in-phase and quadrature amplitudes. Each number may be encoded as one or two fixed-point bytes or four bytes containing a floating-point value, so the sustained data rate may be as low as a few megabytes per second or well over 24 MB/s.

The software processing those samples and displaying the results must maintain that pace, while running as a more-or-less normal program in a desktop operating system. The GNU Radio and GQRX screens in this column came from a Raspberry Pi 3, with four cores running at (up to) 1.2 GHz, which can support support sample rates around 2 MHz.

When GQRX processes incoming data at that pace and produces the waterfalls shown

circuitcellar.com/ccmaterials

RESOURCES

E. Nisley, "WWVB Preamplifier," *Circuit Cellar* 320, 2017.

NIST, Radio Station WWVB, www.nist.gov/time-and-frequency-services/nist-radio-stations/wwvb.

Thumbsat, "The How and Why of Thumbsat," www.thumbsat.com/the-how-and-why-of-thumbsat.

SOURCES

Gqrx SDR Receiver program
Alexandru Csete | http://gqrx.dk/

SMA adapters with integral cables
www.amazon.com/gp/product/B0132N1DM0

SMA adapters
www.amazon.com/dp/B0132MB8LM

Ham-It-Up 125-MHz Upconverter and NESDR Mini 2+
NooElec, Inc. | www.nooelec.com/store/sdr/sdr-receivers/nesdr-mini-2-plus.html | www.nooelec.com/store/sdr/sdr-accessories/rf-blocks/ham-it-up.html

RSP1 SDR
SDRplay | www.sdrplay.com/rsp1/

Thumbnet N3 SDR Receiver
ThumbSat | www.thumbsat.com/thumbnet-n3-sdr-receiver/store-item-1456436783

here, the ARM CPU runs at 40% of maximum capacity. The computations that update the display buffer and shuttle it through the network aren't quite lost in the roundoff, but SDR algorithms account for most of the load. Even a GNU Radio flowgraph that does not produce full-screen graphical output would require about the same amount of computation.

The SDRplay radio's proprietary driver posed several problems, because its software license is incompatible with GPL-based GNU Radio programs and it cannot be distributed with or linked into GNU Radio. Without access to the source code, developers and users have trouble diagnosing problems and can't fix them.

The current version of the Raspbian operating system and the GNU Radio software from its repositories are compatible with the 32-bit ARM CPUs found on earlier Raspberry Pi models. The Pi 3 ARM CPU has a 64 bit architecture and includes floating point hardware, but runs in a 32-bit compatibility mode with those features unavailable.

That produced an interesting failure when I attempted to recompile the osmocom interface with the flag required for the SDRplay driver. Apparently, that option requires a 64-bit instruction unsupported by the Raspian version of the assembler. One workaround requires recompiling the assembler with full support for 64 bit instructions, another involves installing a beta version of Debian, and yet another seems to require compiling and installing a license-washing driver shim between osmocom and the SDRplay driver. I decided to wait for a newer version of Raspbian that could make that problem Go Away.

Conversely, radios based on the RT820/RTL2832 chipset, like the NooElec and ThumbNet SDRs, have GPL drivers, excellent support, and work well. Their ADC dynamic range may require additional feedback from the GNU Radio program, which seems like a reasonable tradeoff.

CONTACT RELEASE

Although practical (that's pronounced "affordable") SDRs can't quite digitize RF directly at the antenna terminals, a small amount of analog hardware can adapt the signal for readily available (pronounced "inexpensive") radios. If you've been wondering what SDR is all about, GNU Radio on a Raspberry Pi 3 will get you into the game, with RF all around you as test signals.

ADuC841
Microcontroller
Design Manual
From Microcontroller Theory to Design Projects
ADuC841 Microcontroller Design Manual
From Microcontroller Theory to Design Projects
If you've ever wanted to design and program with the ADuC841 microcontroller, or other microcontrollers in the 8051 family, this is the book for you. With introductory and advanced labs, you'll soon master the many ways to use a microcontroller. Perfect for academics!
Buy it today!
cc-webshop.com